Transformers without Tears: Improving the Normalization of Self-Attention

Toan Q. Nguyen*
University of Notre Dame

Julian Salazar*
Amazon AWS AI

IWSLT 2019, Hong Kong
paper: https://arxiv.org/pdf/1910.05895.pdf
code: https://github.com/tnq177/transformers_without_tears

*equal contribution

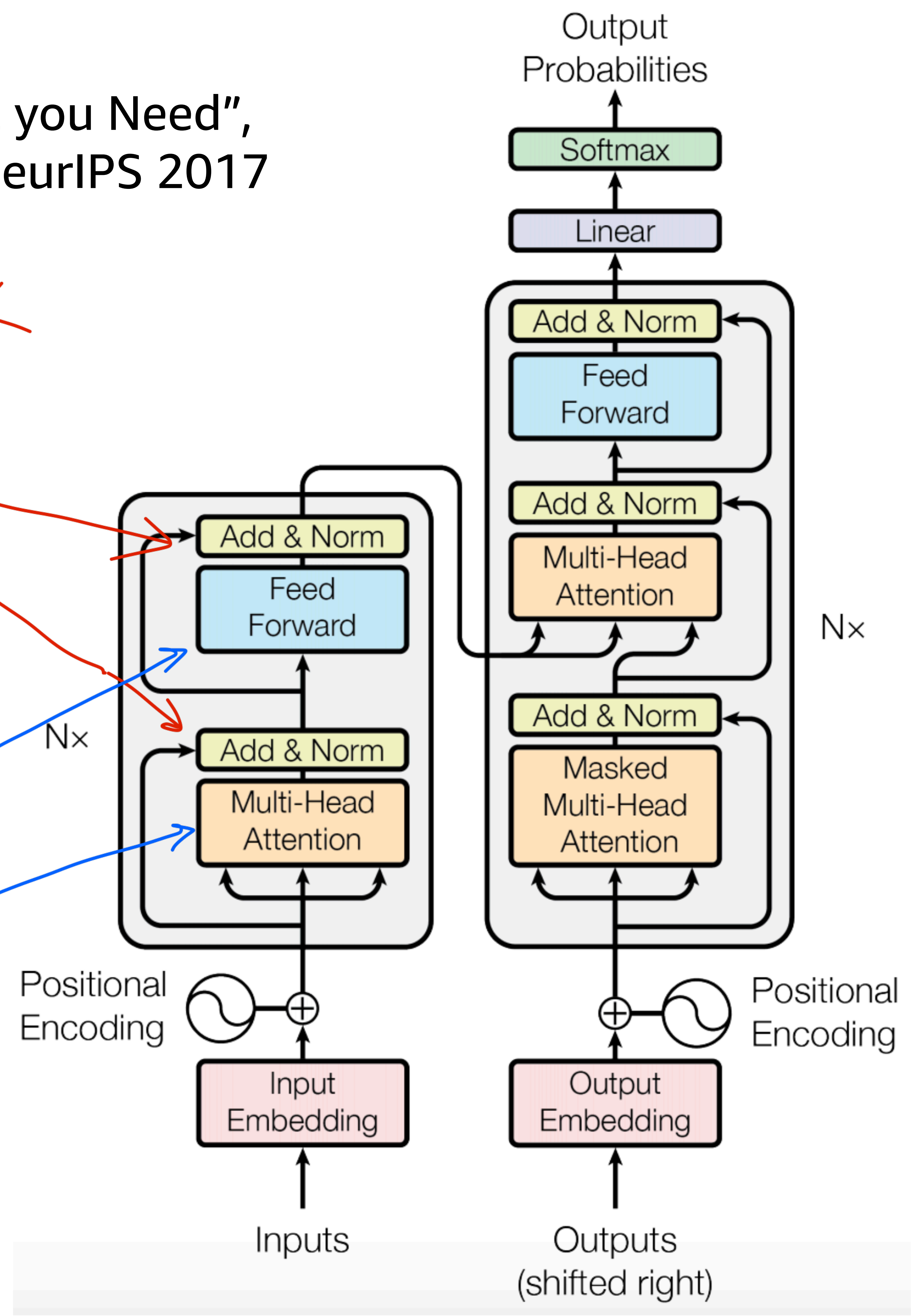
Transformer

“Attention is All you Need”,
Vaswani et al., NeurIPS 2017

3 This talk

2

1

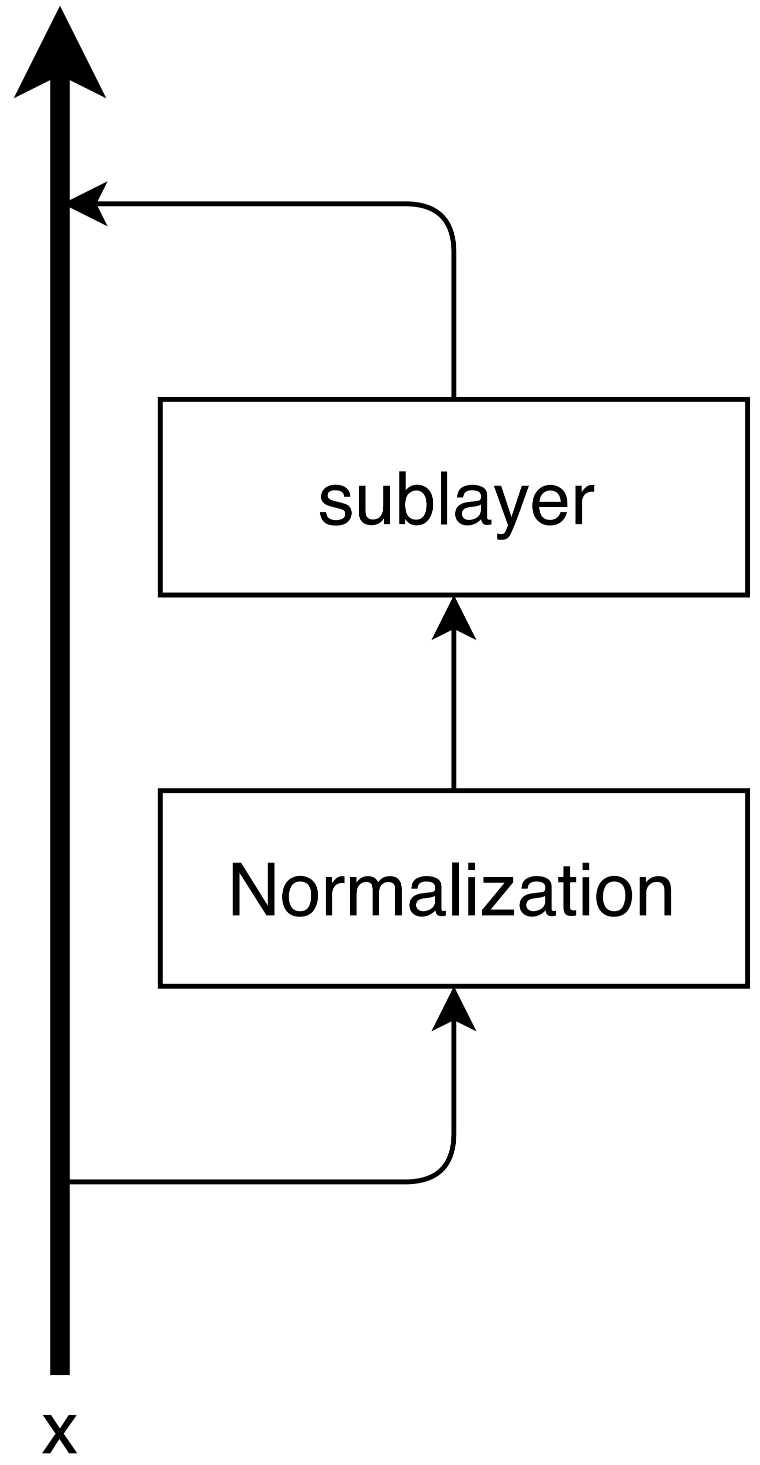


Problems?

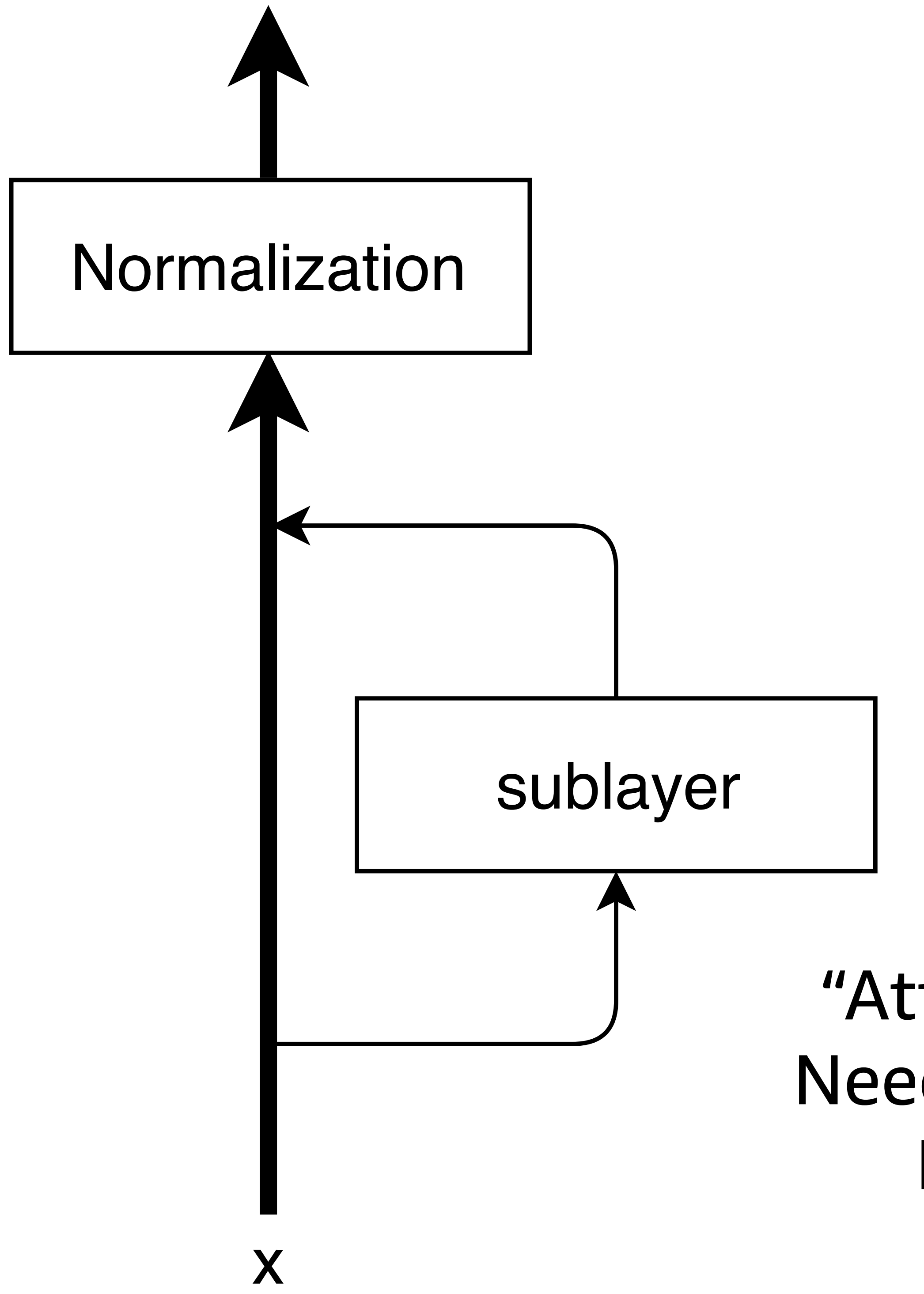
- If you implement your own Transformer, you may find problems with **training stability**:
    - **NO** warmup ==> **NO** convergence
    - **WITH** warmup ==> (sometimes) **NO** convergence
    - (We show the problem lies in the residual connections)
- If you care about **low-resource NMT (...or SLT)**:
    - Previous works on Transformer training focuses on high-resource settings (Vaswani et al. 2017, Shazeer and Stern 2018, Popel and Bojar 2018, Chen et al. 2018…)
    - Can we improve Transformer performance in low-resource NMT?
    - (Yes, via simple changes to normalization)

Stability: PreNorm vs. PostNorm

“Identity Mappings in Deep Residual Networks”, He et al., ECCV 2016



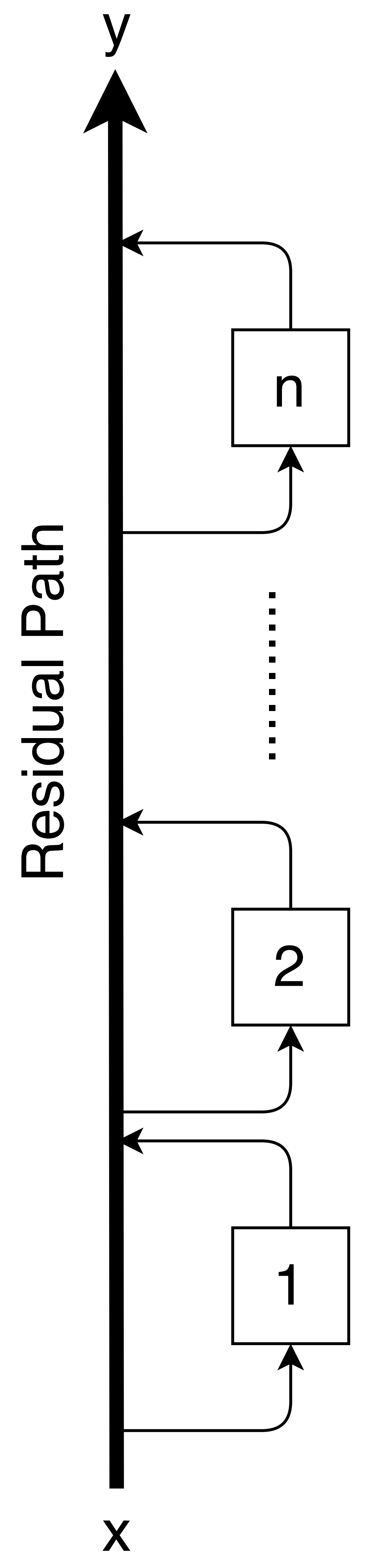
ResNet (PreNorm)



Transformer (PostNorm)

“Attention is All you Need”, Vaswani et al., NeurIPS 2017

PreNorm (ResNet)

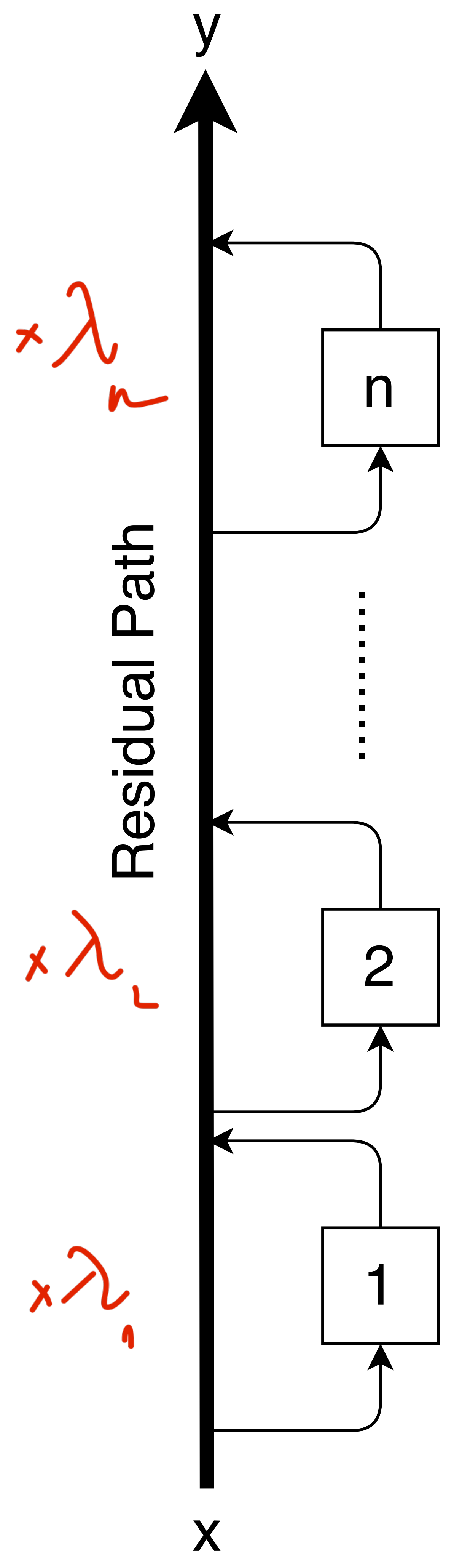


Residual connections
or “identity mappings”

$$x_{l+1} = x_l + F_l(x_l)$$

“contribution” of x_l to $y = x_l$

PreNorm (ResNet)



Suppose we apply λ_l to x_l, i.e.,

$$x_{l+1} = \lambda_l x_l + F_l(x_l)$$

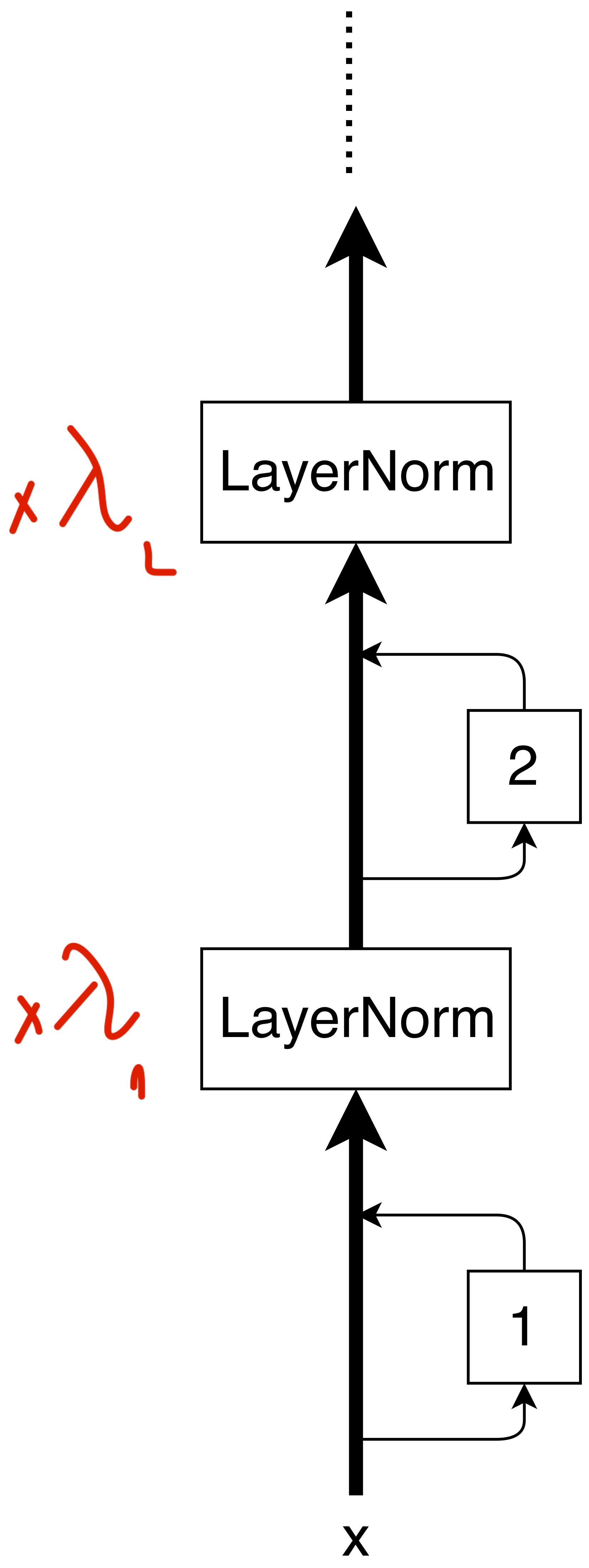
$$\text{“contribution” of } x \text{ to } y \; = \left(\prod_{i=l}^{L-1} \lambda_i \right) x_l$$

$\lambda_i > 1, \; \prod \lambda_i \gg 1 \Rightarrow$ gradient explosion

$\lambda_i < 1, \; \prod \lambda_i \ll 1 \Rightarrow$ gradient vanishing

$\therefore \; \lambda_i$ should always be set to 1 (identity)

PostNorm (Transformer)



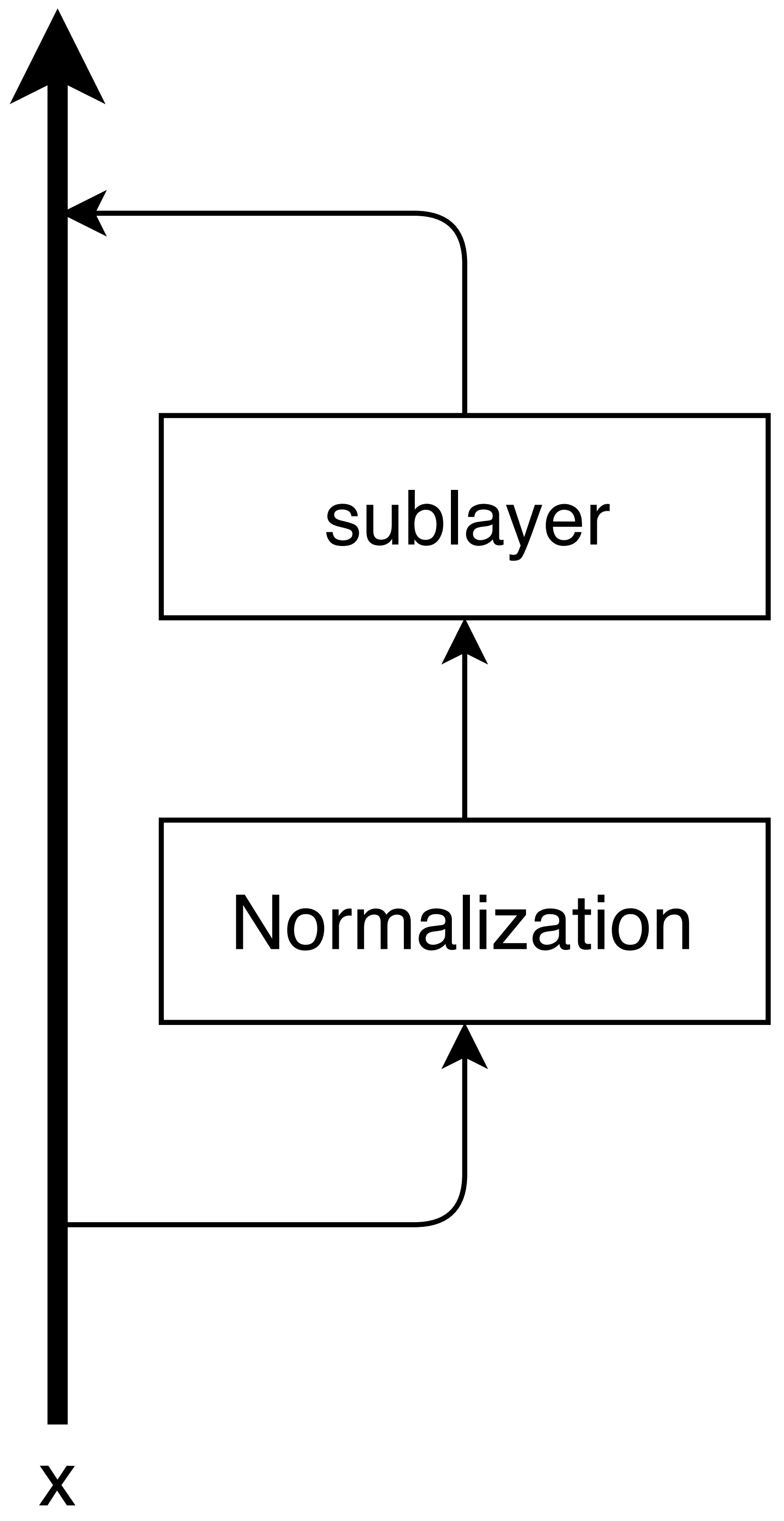
Inserting LayerNorms along the residual path is similar to introducing $\lambda_i \neq 1$, which causes Transformer's instability.

("Learning Deep Transformer Models for Machine Translation", Wang et al., ACL 2019)

Stability: PreNorm vs. PostNorm

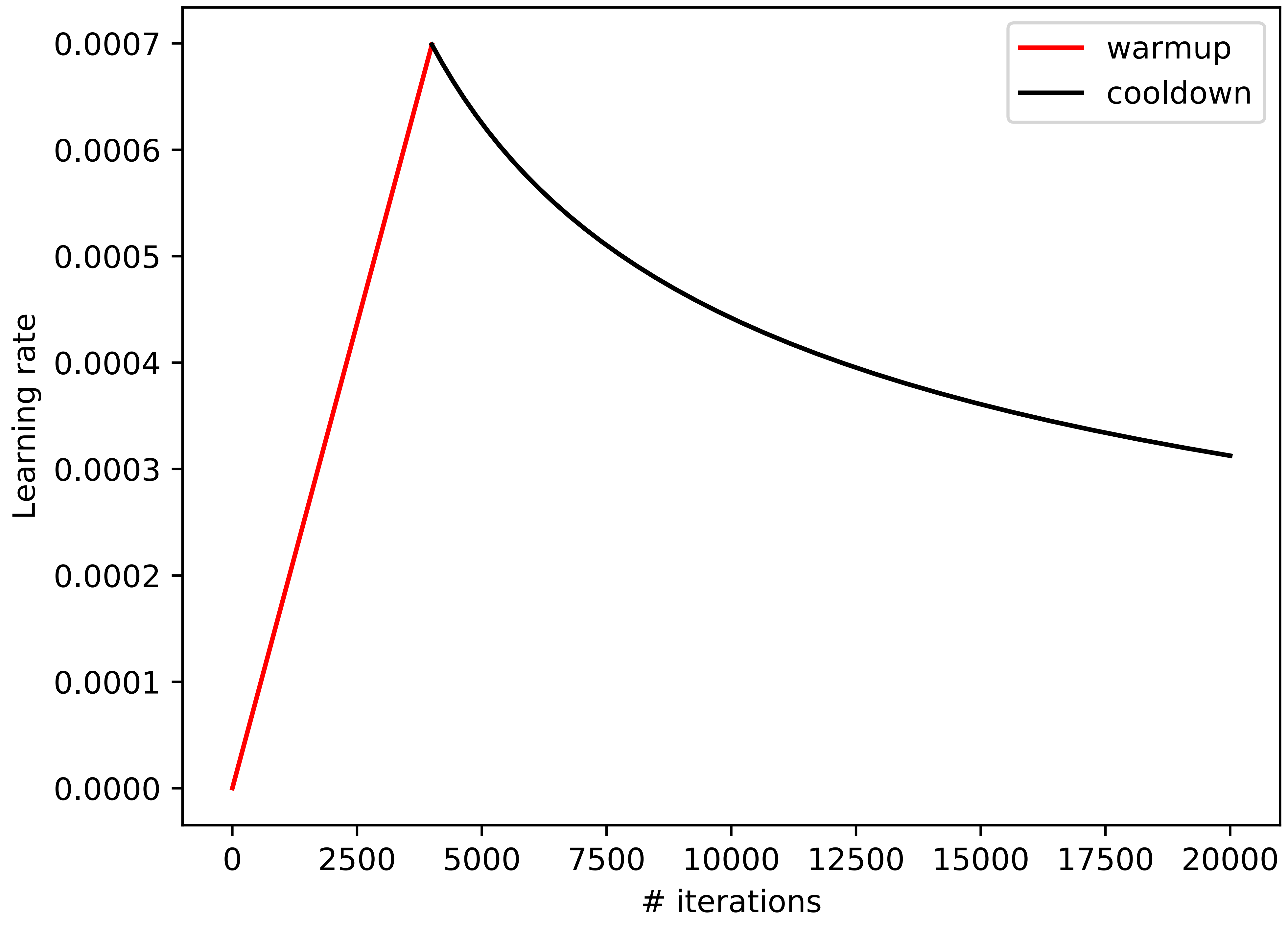
- Mentioned in various works (Chen et al. 2018, Wang et al. 2019, Parisotto et al. 2019)
- Implemented in popular toolkits (tensor2tensor, fairseq, sockeye)
- Discussed by practitioners: https://tunz.kr/post/4, https://github.com/tnq177/witwicky

Their conclusion: **PreNorm allows greater depth and safer training across datasets.**



ResNet (PreNorm)

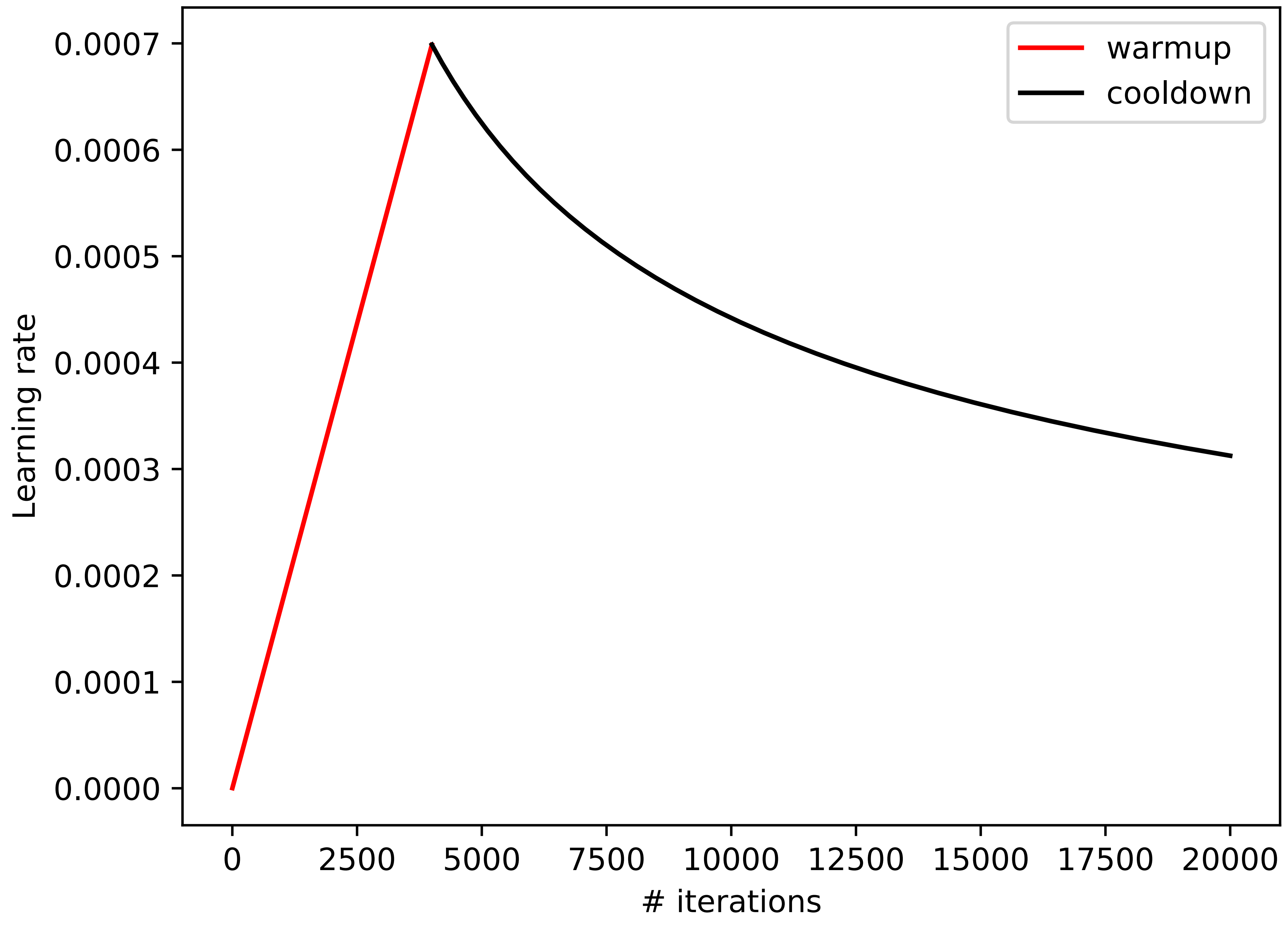
Stability: ...w.r.t. optimization?



Warmup: initial, gradual increase of learning rate.

Empirically tuned.

Stability: ...w.r.t. optimization?



Hypothesis: Warmup is needed to safely stabilize (PostNorm's) LayerNorm gradients.

Stability: Warmup

Xavier normal		# warmup steps 4k	8k	16k
Baseline	PostNorm	fail	fail	5.76
	PreNorm	28.52	28.73	28.32

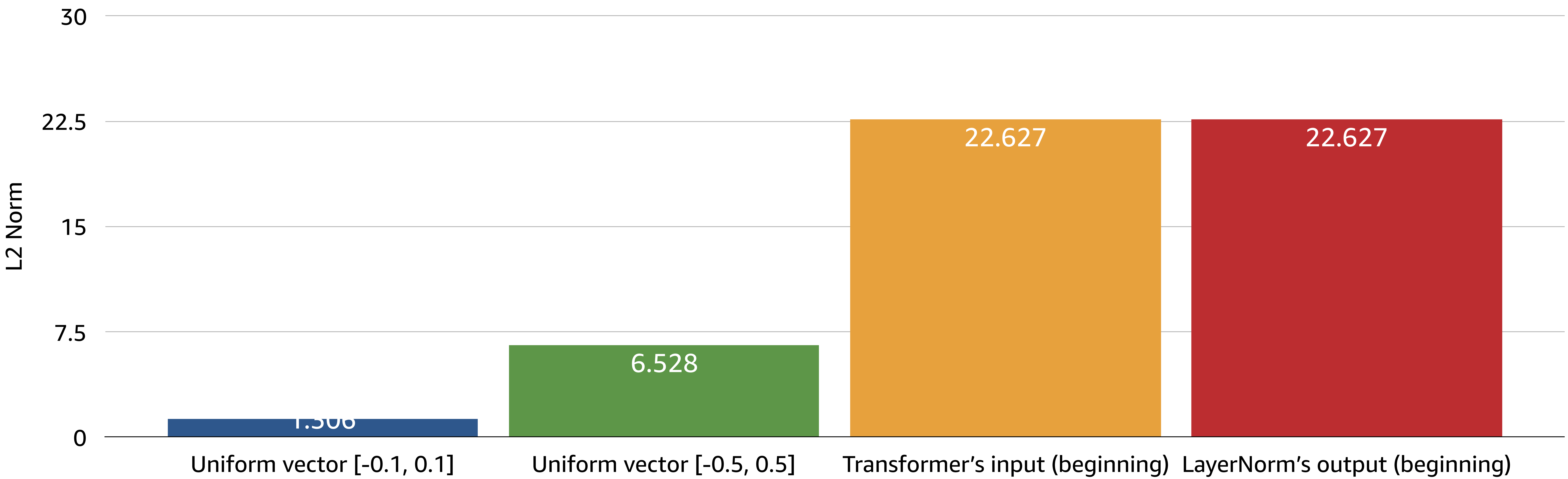
PreNorm works as # warmup steps —> 0!

PostNorm does not. *Can we mitigate its gradients in another way?*

Stability: Weight initialization

In the beginning, Transformer has activations of expected norm $\approx \sqrt{D}$

Idea: Let's shrink the weights to compensate.



Stability: Weight initialization

Attention sublayer's weights: $W_i \sim \mathcal{N}\left(0, \dfrac{2}{D+D}\right)$

Feedforward sublayer's weights: $W_i \sim \mathcal{N}\left(0, \dfrac{2}{D+4D}\right)$

Since the feedforward's weights are smaller, we shrink the attention weights to that (a scale factor of ~0.63).

We propose **SmallInit**: All weights initialized to $W_i \sim \mathcal{N}\left(0, \dfrac{2}{D+4D}\right)$

Stability: Weight initialization

Xavier normal		# warmup steps 4k	8k	16k
Baseline	PostNorm	fail	fail	5.76
	PreNorm	28.52	28.73	28.32
SmallInit	PostNorm	28.17	28.20	28.62
	PreNorm	28.26	28.44	28.33

Table 2: Development BLEU on *en*→*vi* using Xavier normal initialization (baseline versus SmallInit).

Now, PostNorm works as # warmup steps —> 0 too!

SmallInit regains stability for PostNorm. **Let's use it moving forward.**

PreNorm continues to work in both settings.

Stability: Weight initialization

With just one Small(Init) trick:

- PreNorm works as # warmup steps —> 0
- PostNorm works as # warmup steps —> 0

Can we abandon warmup? Stay *tuned.*

Experiments

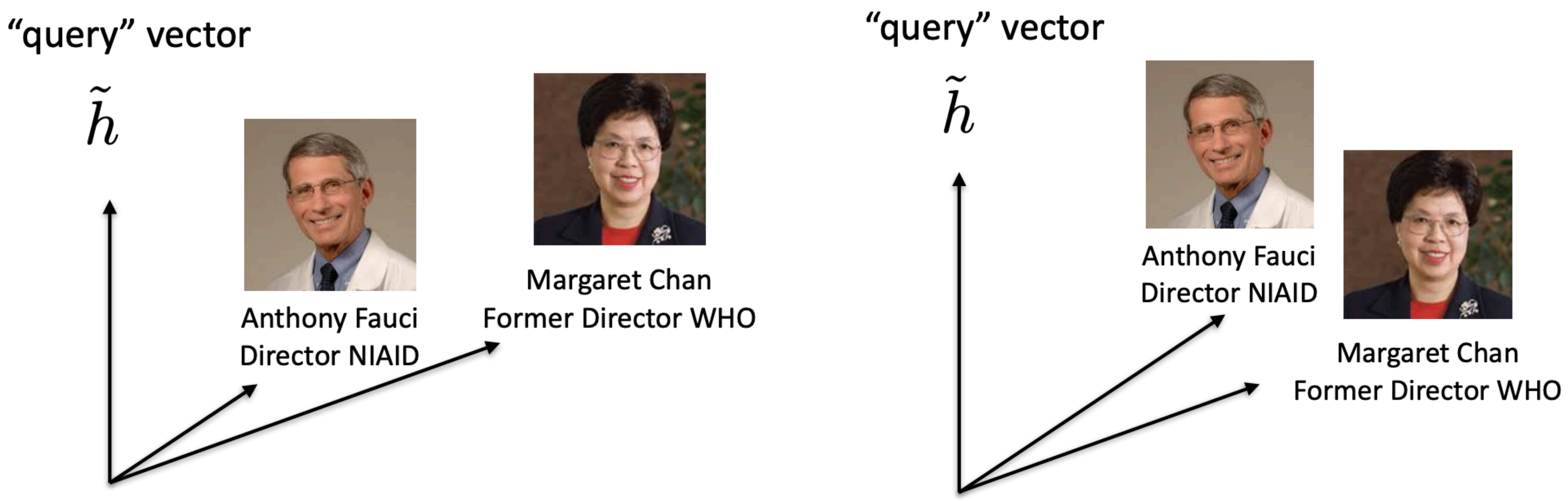
- IWSLT 2015 en->vi, 4 TED Talks pairs from Qi et al., 2018
- Data sizes from 10k to 200+k sentence pairs
- Models are base Transformers
- Joint-language 8k BPE, 8k-step warmup, word dropout, tied input-outputs for strong baselines (+SmallInit so that PostNorm works.)

	gl→en	**sk→en**	***en→vi***	**en→he**	**ar→en**	**average Δ**
PostNorm + LayerNorm (published)	16.2	24.0	29.09	23.66	27.84	-4.05
PostNorm + LayerNorm (1)	18.47	29.37	31.94	27.85	33.39	+0.00
PreNorm + LayerNorm (2)	19.09	29.45	31.92	28.13	33.79	+0.27

Low-resource

- Transformers are **over-parameterized** and **over-confident**
- (e.g., all our models use dropout of 0.3+, label smoothing of 0.1)
- Other works have pruned attention heads, tied self-attention layers, used monolingual pretraining, etc.
- **Can we do something at the normalization steps?**

Low-resource: FixNorm



More frequent words have larger embedding norms than semantically-similar rare words. Here the model mistranslates "Fauci" to "Chan".

Low-resource: FixNorm

Solution: Fix word embedding norm to some g: $e \mapsto g\frac{e}{\|e\|}$ (Nguyen and Chiang, 2018), but with g learnable

	gl→en	sk→en	*en*→*vi*	en→he	ar→en	average Δ
PostNorm + LayerNorm (published)	16.2	24.0	29.09	23.66	27.84	-4.05
PostNorm + LayerNorm (1)	18.47	29.37	31.94	27.85	33.39	+0.00
PreNorm + LayerNorm (2)	19.09	29.45	31.92	28.13	33.79	+0.27
PreNorm + FixNorm + LayerNorm (3)	19.38	29.50	32.45	28.39	34.35†	+0.61

Low-resource: Layer normalization

LayerNorm (Ba et al., 2016) stems from BatchNorm (Ioffe and Szegedy, 2015)

Ioffe and Szegedy, 2015: BatchNorm helps by solving the internal covariate shift

Santurkar et al., 2018: BatchNorm actually helps by smoothing the loss landscape. e.g., **normalizing by other statistics work too**

Zhang and Sennrich, 2019: propose RMSNorm which normalizes by root mean square. It’s **faster** than LayerNorm and achieves **comparable** results

Low-resource: ScaleNorm

LayerNorm: $\bar{x}_i = \frac{x_i - \mu}{\sigma} a_i + b_i$

RMSNorm: $\bar{x}_i = \frac{x_i}{\text{RMS(x)}} a_i$, $\text{RMS(x)} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2}$

We propose **ScaleNorm:** $\bar{x} = g \frac{x}{\|x\|}$

Low-resource: ScaleNorm

ScaleNorm is similar to FixNorm but on the inputs, not on the embedding

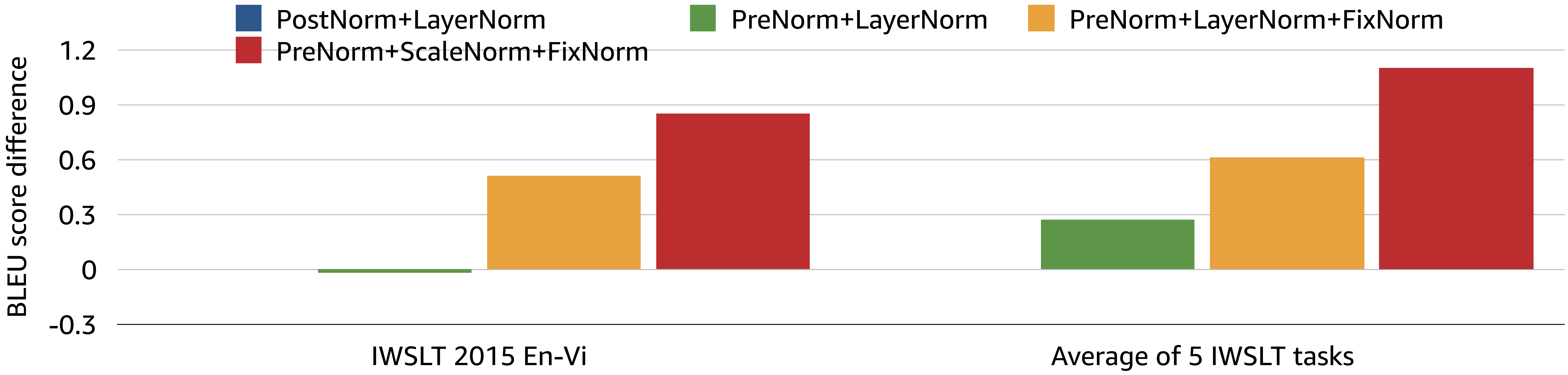
ScaleNorm has **no centering, no mean-shifting after scaling, 1 scale parameter per layer**

Speed: ScaleNorm > RMSNorm > LayerNorm

ScaleNorm+FixNorm at final output layer = maximizing cosine distance

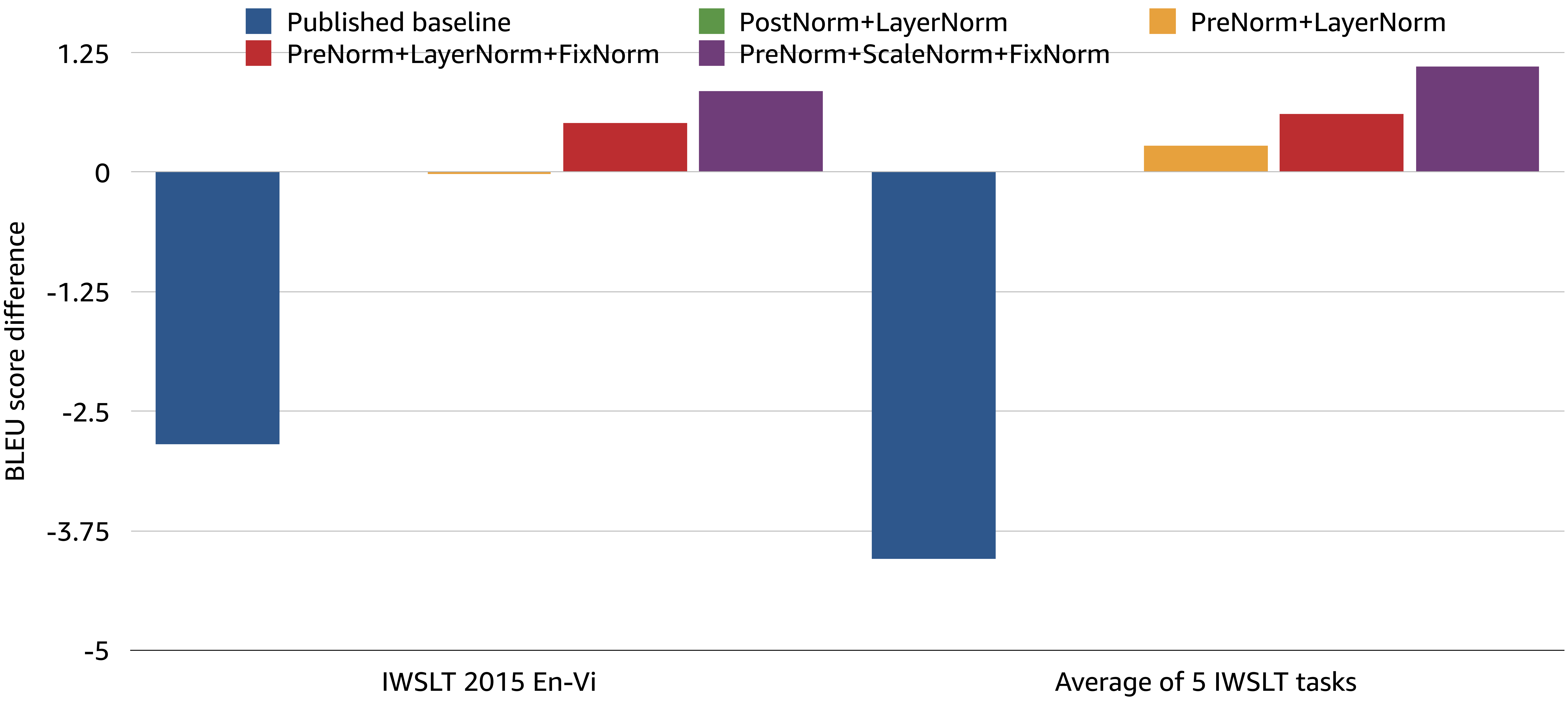
Nguyen and Chiang (2018) used a fixed g for ScaleNorm+FixNorm which improved low-resource NMT

Experiments



	gl→en	sk→en	*en→vi*	en→he	ar→en	average Δ
PostNorm + LayerNorm (published)	16.2	24.0	29.09	23.66	27.84	-4.05
PostNorm + LayerNorm (1)	18.47	29.37	31.94	27.85	33.39	+0.00
PreNorm + LayerNorm (2)	19.09	29.45	31.92	28.13	33.79	+0.27
PreNorm + FixNorm + LayerNorm (3)	19.38	29.50	32.45	28.39	34.35†	+0.61
PreNorm + FixNorm + ScaleNorm (4)	20.91‡*	30.25‡*	32.79*	28.44*	34.15*	+1.10

Experiments



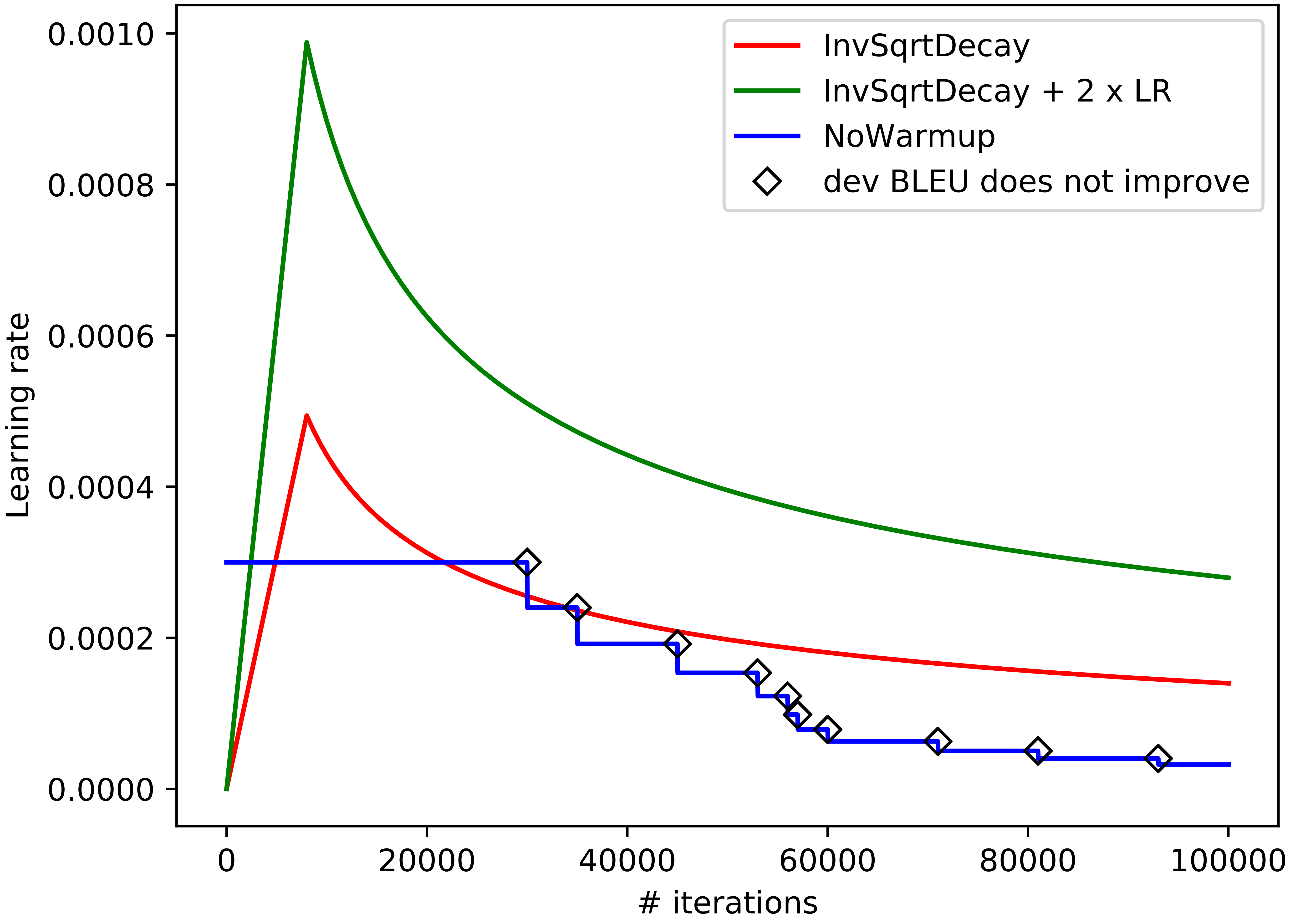
Experiments: Learning rate

Do we really need warmup?

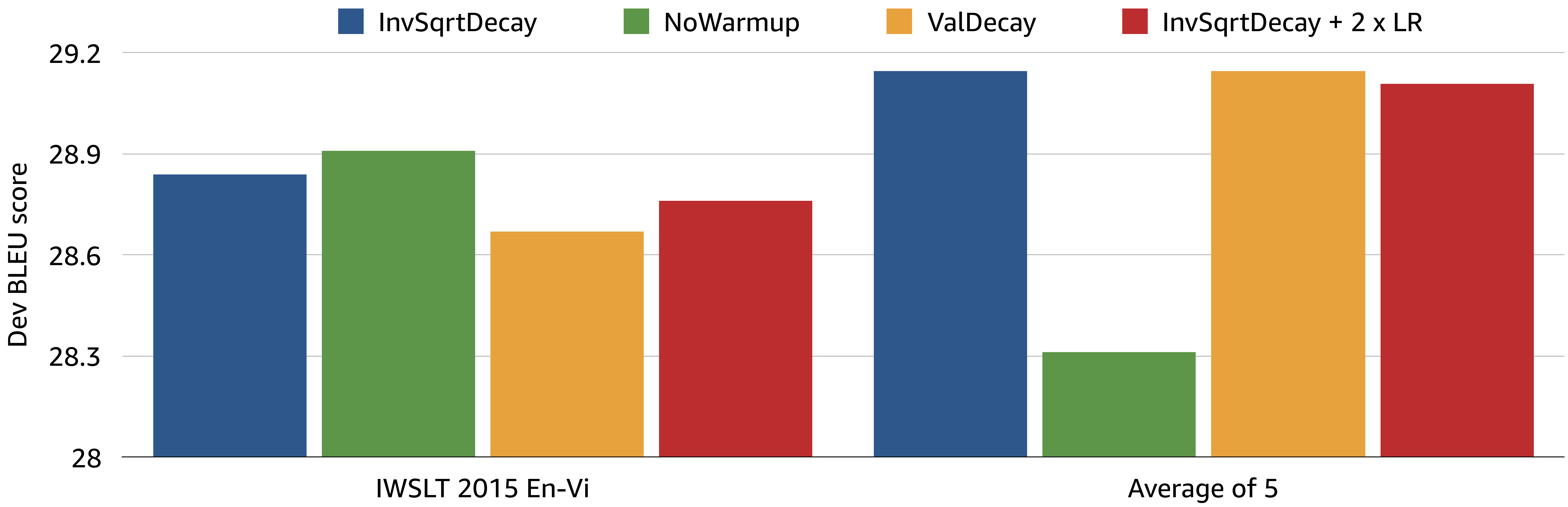
Does the good old “decay when dev BLEU doesn’t improve” still work?

Can we train low-resource on small batch sizes (4096 tokens/batch) with very high learning rate?

Experiments: Learning rate



Experiments: Learning rate



(we can often get away without warmup, but warmup is still useful)

Experiments: Learning rate

Do we really need warmup? No!

Does the good old "decay when dev BLEU doesn't improve" still work? Yes!

Can we train low-resource on small batch sizes (4096 tokens/batch) with very high learning rate? Yes!

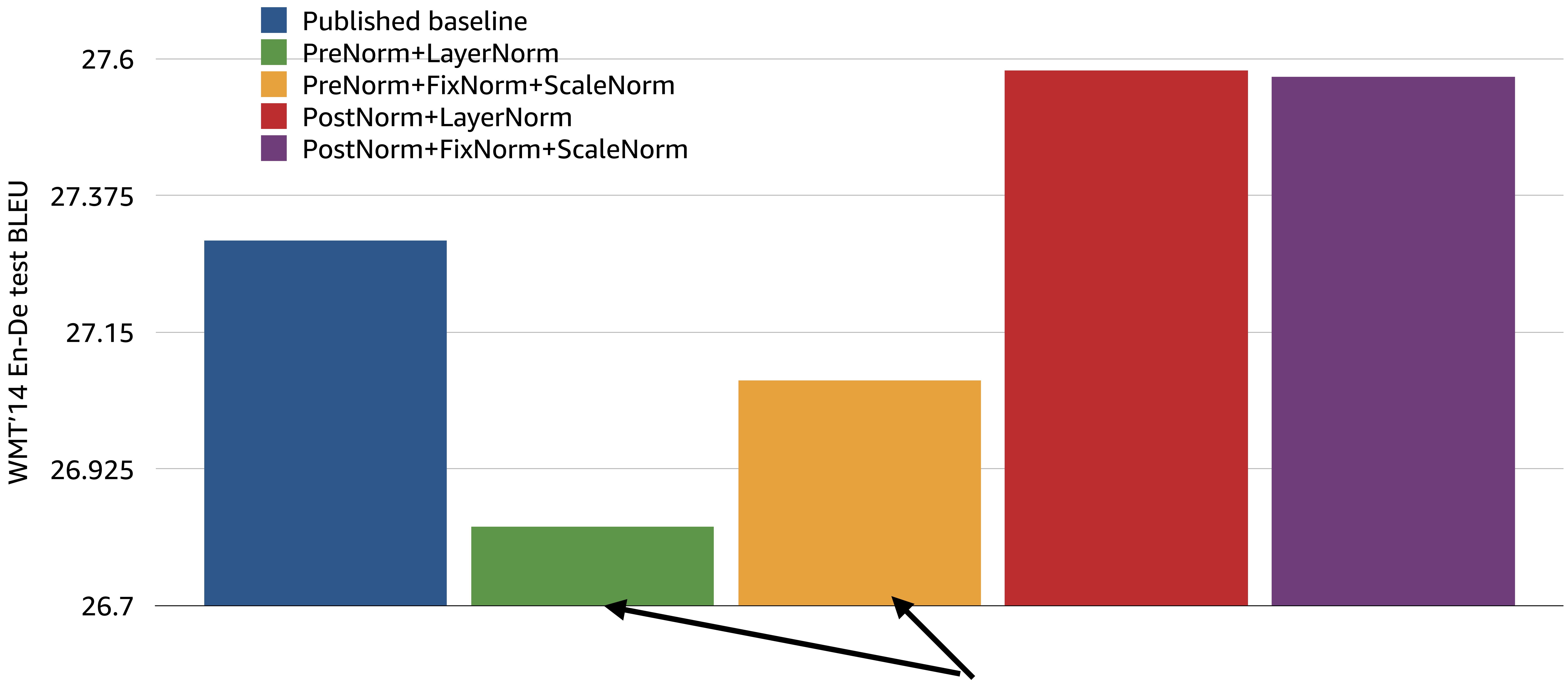
Experiments: PreNorm vs. PostNorm (again)

	4 layers	5 layers	6 layers
POSTNORM	18.31	fails	fails
PRENORM	28.33	28.13	28.32

Table 5: Development BLEU on *en→vi* using NOWARMUP, as number of encoder/decoder layers increases.

Can SmallInit help PostNorm without warmup? No :(

High resource (a different story)



We use WMT 2014 en-de, via fairseq. PreNorm degrades performance!

High resource (a different story)

High-resource often uses large batch size which has more stable gradients.
This could help solving the instability problem.

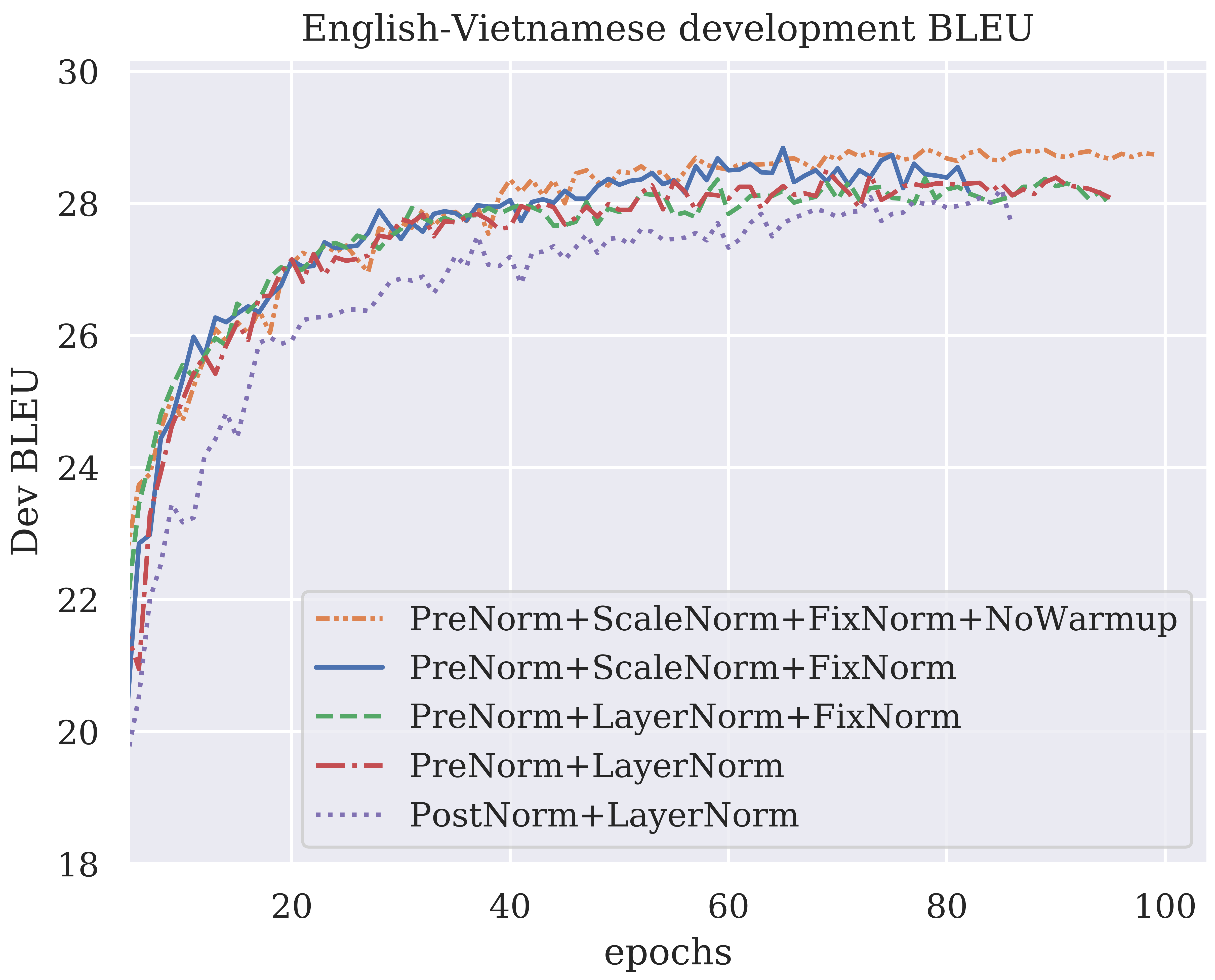
ScaleNorm + FixNorm achieves comparable results

ScaleNorm is faster than LayerNorm

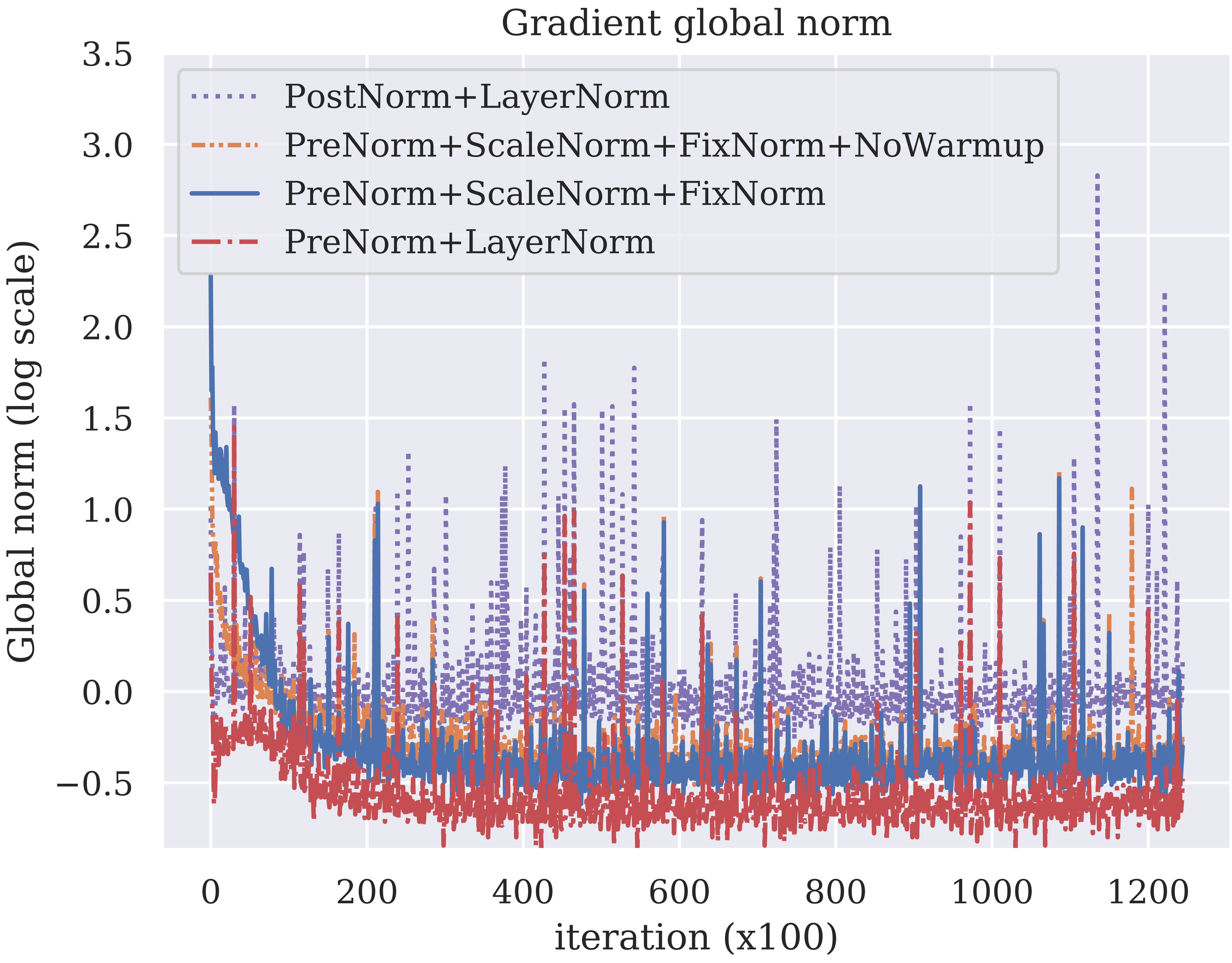
We recommend always replacing LayerNorm with ScaleNorm+FixNorm

(The PreNorm vs. PostNorm story is not finished!)

Analysis: Perfomance curves

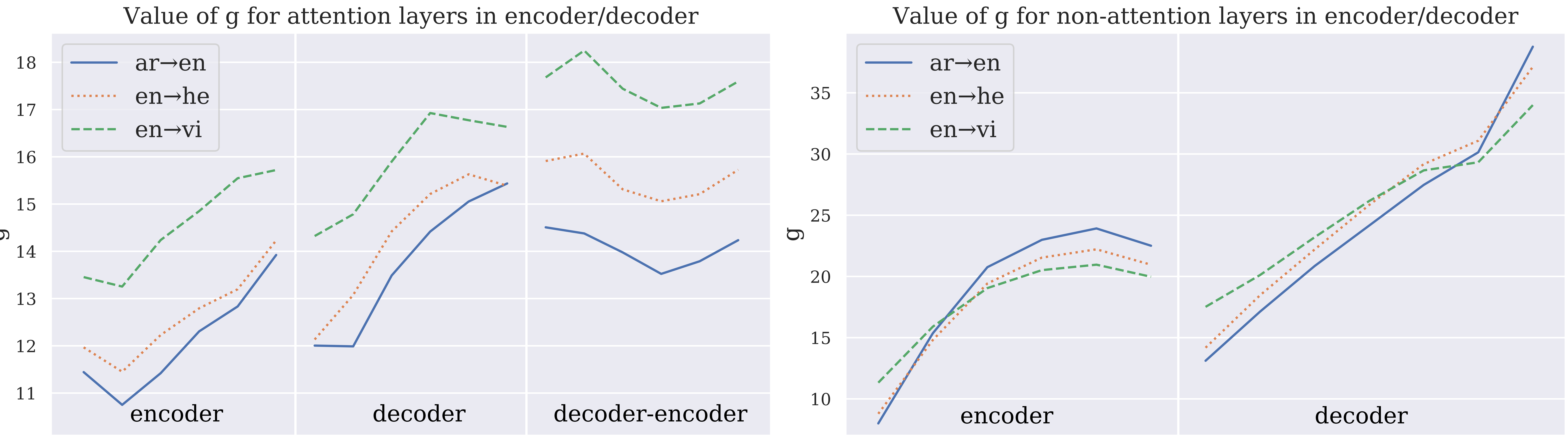


Analysis: Gradients



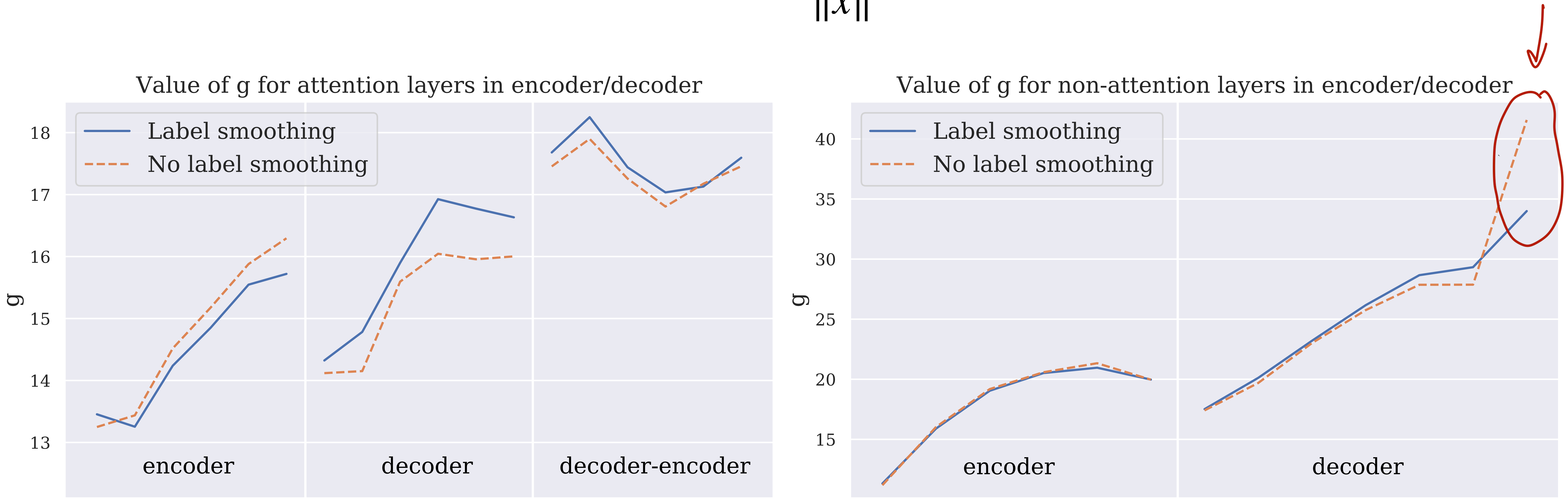
Analysis: Learned g-values

$$\bar{x} = g\frac{x}{\|x\|}$$



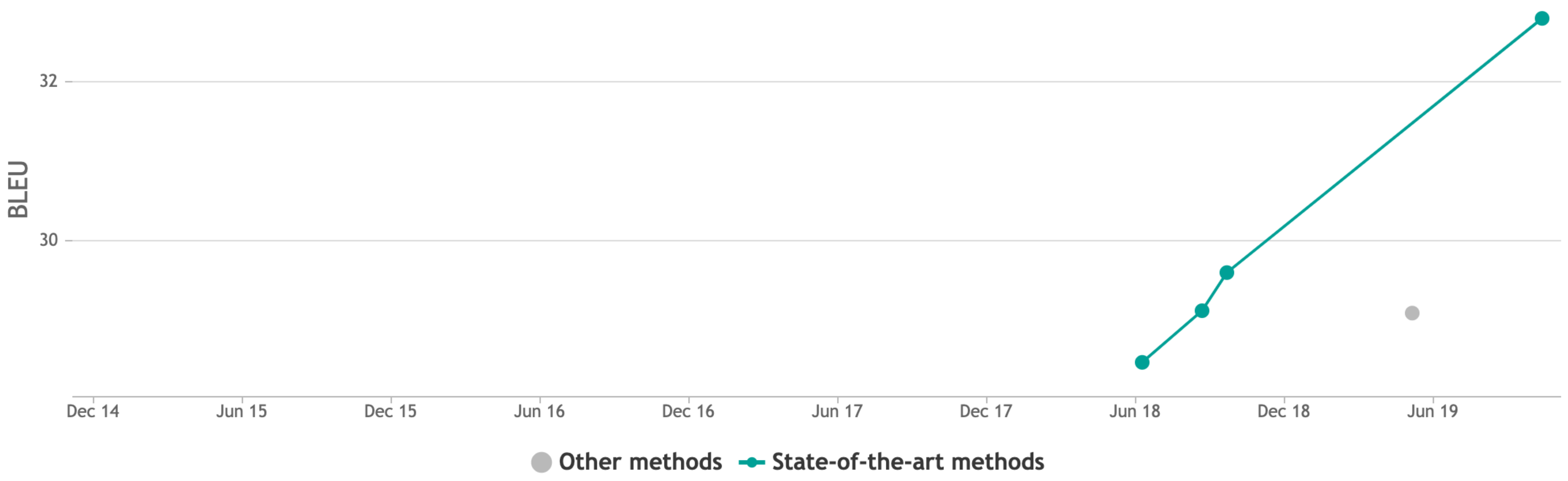
Analysis: Label smoothing

$$\bar{x} = g \frac{x}{\|x\|}$$



Conclusion

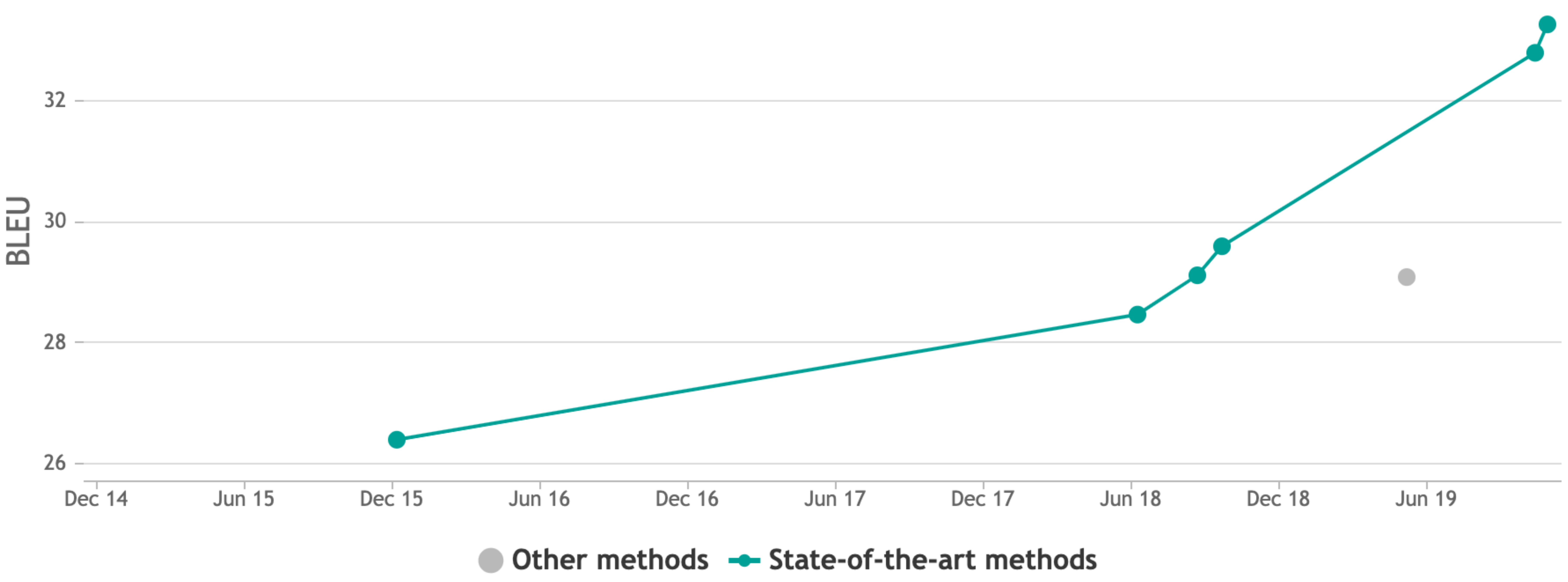
- We propose 3 changes to Transformer: PreNorm + FixNorm + ScaleNorm
- Significantly improves low-resource, Transformer-based NMT
- Comparable on high-resource NMT (FixNorm+ScaleNorm)
- Faster



Edit

RANK	METHOD	BLEU	PAPER TITLE	YEAR	PAPER	CODE
1	Transformer+BPE+FixNorm+ScaleNorm	32.8	Transformers without Tears: Improving the Normalization of Self-Attention	2019		

https://paperswithcode.com/sota/machine-translation-on-iwslt2015-english-1



Edit

RANK	METHOD	BLEU	PAPER TITLE	YEAR	PAPER	CODE
1	Transformer+BPE-dropout	33.27	BPE-Dropout: Simple and Effective Subword Regularization	2019		

https://paperswithcode.com/sota/machine-translation-on-iwslt2015-english-1

Questions?

paper: https://arxiv.org/pdf/1910.05895.pdf

code: https://github.com/tnq177/transformers_without_tears

References

1. “Attention is all you need”, Vaswani et al., 2017
2. “Adafactor: Adaptive Learning Rates with Sublinear Memory Cost”, Shazeer and Stern, 2018
3. “Training tips for the Transformer model”, Popel and Bojar, 2018
4. “The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation”, Chen et al., 2018
5. “Identity Mappings in Deep Residual Networks”, He et al., 2016
6. “Learning Deep Transformer Models for Machine Translation”, Wang et al., 2019
7. “On Layer Normalization in the Transformer architecture”, Anonymous, 2019
8. “Stabilizing Transformers for Reinforcement Learning”, Parisotto et al., 2019
9. “The Sockeye neural machine translation toolkit”, Hieber et al., 2018
10. “Tensor2Tensor for Neural Machine Translation”, Vaswani et al., 2018
11. “fairseq: A Fast, Extensible Toolkit for Sequence Modeling”, Ott et al., 2019

References

12. “Improving Lexical Choice in Neural Machine Translation”, Nguyen and Chiang, 2018
13. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”, Ioffe and Szegedy, 2015
14. “How does batch normalization help optimization?”, Santurkar et al., 2018
15. “Layer Normalization”, Ba et al., 2016
16. “Root Mean Square Layer Normalization”, Zhang and Sennrich, 2019
17. “Effective approaches to attention-based neural machine translation”, Luong et al., 2015
18. “Deep Sparse Rectifier Neural Networks”, Glorot et al., 2011
19. “Deep Residual Learning for Image Recognition”, He et al., 2015
20. “Residual Networks Behave Like Ensembles of Relatively Shallow Networks”, Veit et al., 2016
21. “When and Why are pre-trained word embeddings useful for Neural Machine Translation”, Qi et al., 2018